Kapitel 45
Installation und erste App

Um Apps schreiben zu können, benötigen Sie zusätzlich zum Java-SDK noch

- das **Android-SDK**
- **Eclipse** mit dem **Android-Plug-In** (optional, aber empfehlenswert)
- ein Android-Smartphone (optional)

Eclipse finden Sie übrigens auf der Buch-DVD. Wir hätten gerne noch ein kostenloses Android-Smartphone hinzugepackt, aber es ließ sich leider kein Sponsor finden.

DVD

45.1 Das Android-SDK

Das Android-SDK finden Sie aus den oben genannten Gründen nicht auf der Buch-DVD, Sie müssen es von der Android-Website herunterladen: `http://developer.android.com/sdk/index.html`.

SDK und Plug-In werden derzeit von Google in erschreckend kurzen Produktzyklen aktualisiert. Während wir beim Schreiben dieses Buches also noch mit dem Android SDK 12 (dem damals aktuellen SDK) gearbeitet haben, werden Sie mit ziemlicher Sicherheit schon mit dem SDK 13, 14 oder noch höher arbeiten. Lassen Sie sich daher nicht zu sehr verwirren, wenn eine Abbildung zur Eclipse-Android-Oberfläche einmal nicht hundertprozentig mit Ihrem Bildschirm übereinstimmt oder ein Eclipse-Android-Menübefehl sich geändert hat.

STOP

Während wir beim Schreiben des Buches allerdings noch die Version 11 benutzten, berücksichtigt der vorliegende Text die Version 16, also

- installer_r16-windows.exe bzw.
- android-sdk_r16-linux.tgz

INFO

Die für die Installation bedeutendste Änderung ist wohl, dass der SDK Manager und der AVD Manager nicht mehr wie im Buch beschrieben in einem Tool-Programm vereinigt sind, sondern nun eigenständige Programme sind.

45.1.1 Installation

Die Installation des Android-SDK ist schnell erledigt.

1. Laden Sie die Setupdatei von der Android-Website `http://developer.android.com/sdk/index.html` herunter und führen Sie sie aus.

   Windows: Für Windows hieß die Setupdatei bisher üblicherweise *installer_rNN-windows.exe*, wobei »rNN« für die aktuelle Versionsnummer steht. Zum Installieren doppelklicken Sie auf die heruntergeladene Setupdatei.

   Linux: Für Linux hieß die Setupdatei bisher üblicherweise *android-sdk_rNN-linux_x86.tgz*, wobei »rNN« für die aktuelle Versionsnummer steht. Kopieren Sie die Datei in das übergeordnete Verzeichnis, unter dem das Android-SDK installiert werden soll, und entpacken Sie die Datei mit dem Befehl:

   ```
   tar -xzf android-sdk_rNN-linux_x86.tgz
   ```

   Danach können Sie die ZIP-Datei löschen.
2. Folgen Sie den Anweisungen des Setupprogramms und lassen Sie das SDK installieren.

   Windows: Eigentlich müssen Sie sich nur mittels der NEXT-Schaltfläche durch die Dialoge klicken.

   In früheren Versionen gab es das Problem, dass das Setupprogramm im ersten Anlauf das Java-JDK nicht finden konnte. Mittlerweile taucht dieser Fehler unserer Erfahrung nach nicht mehr auf. Falls doch: Drücken Sie einfach die Schaltfläche BACK, um einen Dialog zurückzugehen, und anschließend wieder NEXT. Danach sollte der Dialog aus Abbildung 1 zu sehen sein, der bestätigt, dass das JDK gefunden wurde.

   Im weiteren Verlauf können Sie das Installationsverzeichnis auswählen. Kontrollieren Sie dabei den vorgeschlagenen Installationsort. Dieser enthält unter Umständen Leerzeichen im Pfad und auf manchen Windows-Architekturen führt dies dazu, dass das SDK später nicht vom Eclipse-Android-Plug-In verwendet werden kann. Wählen Sie ein Verzeichnis, in dessen Pfad keine Leer- oder Sonderzeichen vorkommen, wie z.B. *C:\Compiler\Android\android-sdk*.

   Deaktivieren Sie auf der letzten Seite die Option zum Starten des SDK-Managers, bevor Sie auf FINISH klicken.

   Linux: Unter Linux entfällt dieser Schritt.

Android SDK Tools Setup
Java SE Development Kit
Detect whether Java SE Development Kit is installed.
Android SDK relies on the Java SE Development Kit (JDK).
Java SE Development Kit (JDK) version 1.7 has been found.
Log
Nullsoft Install System v24-Nov-2011.cvs
< Back
Next >
Cancel

Abbildung 45.1: Das Setupprogramm hat eine installierte JDK-Version gefunden.

Der erste Teil des Android-SDK ist nun eingerichtet, doch es fehlen noch diverse Komponenten: wie z.B. die Bibliotheken, die Sie dringend für die Programmierarbeit benötigen. Mithilfe des SDK-Managers können Sie diese nun installieren.

Der SDK-Manager benötigt eine bestehende Internetverbindung!

STOP

3. Starten Sie den SDK-Manager.
   Windows: Öffnen Sie dazu z.B. den Windows Explorer, wechseln Sie in das Verzeichnis des Android-SDK und starten Sie den SDK-Manager durch Doppelklick auf die Datei *SDK Manager.exe*. (Wenn Sie nicht als Administrator angemeldet sind, klicken Sie mit der rechten Maustaste auf die Datei und wählen Sie im Kontextmenü den Befehl Als Administrator ausführen aus.)
   Linux: Wechseln Sie in das *tools*-Verzeichnis des Android-SDK und starten Sie das Programm *android*..

In dem Fenster Choose Packages to Install sehen Sie nun links eine größeren Auswahl an Komponenten (Packages), die Sie installieren können. Die Komponenten mit dem grünen Häkchen vor dem Namen sind für die Installation vorgemerkt.

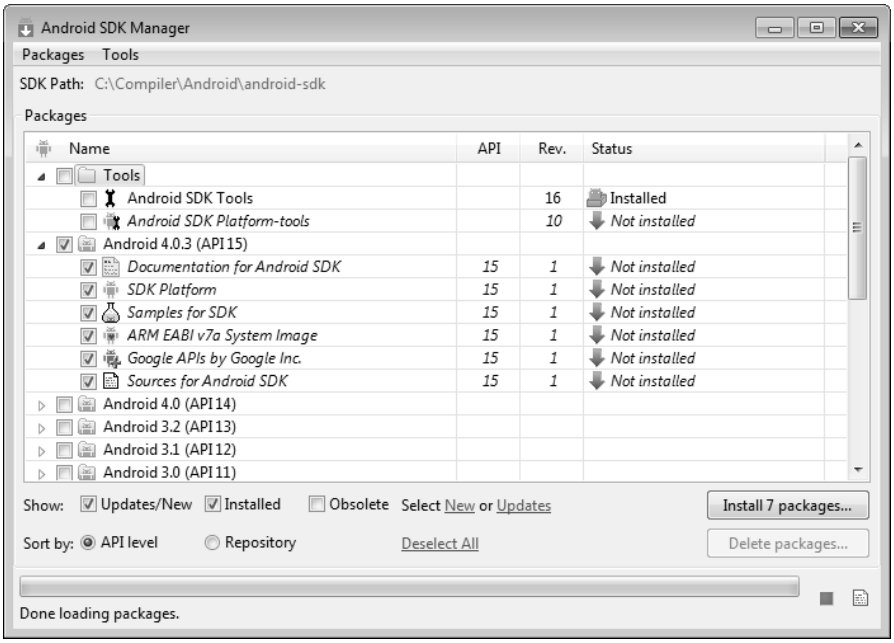

Android SDK Manager
Packages Tools
SDK Path: C:\Compiler\Android\android-sdk
Packages
Name API Rev. Status
Tools
Android SDK Tools 16 Installed
Android SDK Platform-tools 10 Not installed
Android 4.0.3 (API 15)
Documentation for Android SDK 15 1 Not installed
SDK Platform 15 1 Not installed
Samples for SDK 15 1 Not installed
ARM EABI v7a System Image 15 1 Not installed
Google APIs by Google Inc. 15 1 Not installed
Sources for Android SDK 15 1 Not installed
Android 4.0 (API 14)
Android 3.2 (API 13)
Android 3.1 (API 12)
Android 3.0 (API 11)
Show: Updates/New Installed Obsolete Select New or Updates
Install 7 packages...
Sort by: API level Repository
Deselect All
Delete packages...
Done loading packages.

Abbildung 45.2: Der SDK-Manager hat für Sie eine Vorauswahl der Komponenten getroffen, die noch installiert werden sollten.

4. Markieren Sie zusätzlich das Kästchen für die Plattformen Android 2.2, mit der wir in diesem Buch standardmäßig arbeiten.

EXKURS

Plattformen und APIs

Wie von jeder Software kommen auch von dem Android-Betriebssystem ständig neue und erweiterte Versionen (Plattformen) heraus. Zu jeder dieser Plattformen gibt es eine eigene API. Der SDK-Manager lädt per Voreinstellung die Bibliotheken zu allen Android-Versionen herunter, sodass Sie später, wenn Sie beginnen, eine App zu schreiben, frei auswählen können, zu welcher Android-Version die App kompatibel sein soll.

5. Klicken Sie auf die INSTALL...-Schaltfläche.
6. Im nachgeschalteten Dialog CHOOSE PACKAGES TO INSTALL aktivieren Sie die Option ACCEPT ALL, um für alle zu installierenden Komponenten die Lizenzvereinbarung anzunehmen (soweit es solche gibt) und klicken Sie dann auf INSTALL, um die ausgewählten Komponenten installieren zu lassen.

Das Herunterladen und Installieren der Komponenten wird vermutlich etwas länger dauern. Haben Sie also ein wenig Geduld. Im automatisch aufspringenden Log-Fenster bzw. anhand des Fortschrittsbalkens im SDK Manager-Fenster können Sie das Fortschreiten der Installation verfolgen. Geht alles gut, werden die installierten Komponenten angezeigt.

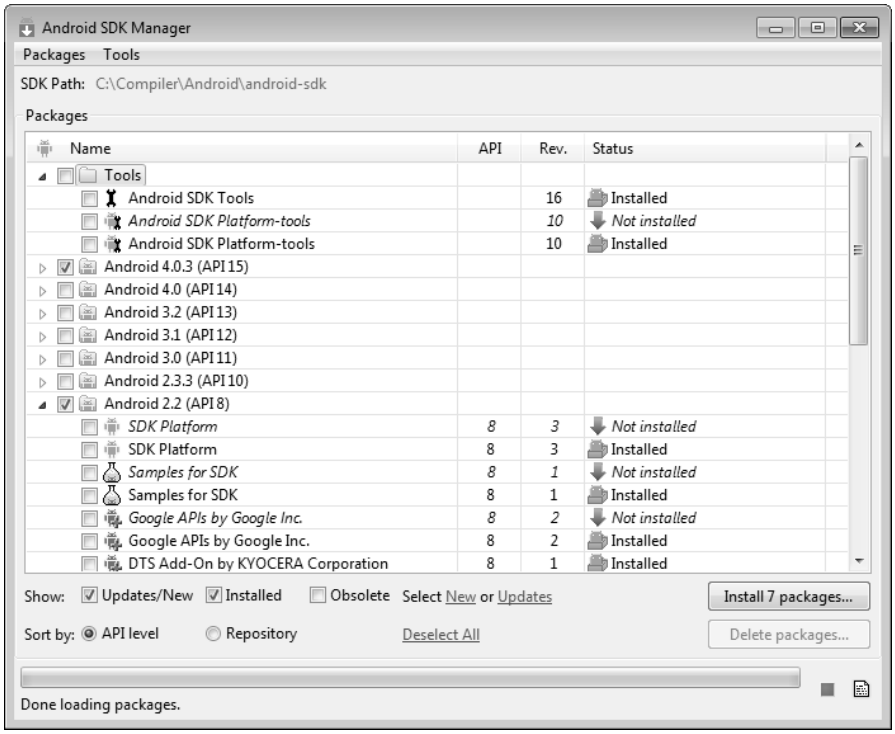

Android SDK Manager
Packages Tools
SDK Path: C:\Compiler\Android\android-sdk
Packages
Name API Rev. Status
Tools
Android SDK Tools 16 Installed
Android SDK Platform-tools 10 Not installed
Android SDK Platform-tools 10 Installed
Android 4.0.3 (API 15)
Android 4.0 (API 14)
Android 3.2 (API 13)
Android 3.1 (API 12)
Android 3.0 (API 11)
Android 2.3.3 (API 10)
Android 2.2 (API 8)
SDK Platform 8 3 Not installed
SDK Platform 8 3 Installed
Samples for SDK 8 1 Not installed
Samples for SDK 8 1 Installed
Google APIs by Google Inc. 8 2 Not installed
Google APIs by Google Inc. 8 2 Installed
DTS Add-On by KYOCERA Corporation 8 1 Installed
Show: Updates/New Installed Obsolete Select New or Updates
Install 7 packages...
Sort by: API level Repository
Deselect All
Delete packages...
Done loading packages.

Abbildung 45.3: Die für die Installation vorgesehenden Pakete (Status *Not installed*) wurden installiert (gedoppelter Eintrag mit Status *Installed*). Nach einen Neustart des SDK Managers werden die veralteteten *Not Installed*-Einträge nicht mehr angezeigt.

7. Schließen Sie die Fenster des SDK-Managers.

Deinstallation

Wenn Sie das Android-SDK irgendwann wieder deinstallieren möchten, wechseln Sie in das Installationsverzeichnis, Unterzeichnis *android-sdk* und rufen Sie dort das Programm *uninstall.exe* auf. Anschließend können Sie von Hand noch das Installationsverzeichnis selbst löschen.

Linux-Anwender löschen einfach das Verzeichnis der Android-Installation.

45.1.2 Dokumentation und API-Referenz

Werfen Sie einen Blick in das Installationsverzeichnis des Android-SDK. Neben den Plattform-Bibliotheken, Tools und Beispiel-Apps finden Sie dort auch die komplette Android-Dokumentation (Unterverzeichnis *docs*) – ein Abbild der Android-Website.

Tabelle 45.1: Wichtige Seiten der Android-Dokumentation

Seite	Inhalt
SDK	Hier können Sie sich über die Besonderheiten der diversen Android-Plattformen informieren. Über den Link *SDK System Requirements*, rechts unten im Abschnitt *More Information*, können Sie die Systemvoraussetzungen einsehen, die für eine erfolgreiche Installation des Android-SDK Bedingung sind.
Dev Guide	Hier finden Sie Hinweise, Artikel und Tutorien zu praktisch allen Aspekten der Android-Programmierung.
Reference	Dies ist die Referenz der Elemente aus der Android-Bibliothek. Die Referenz ist analog der Java-API-Dokumentation aufgebaut.

45.2 Eclipse

Eclipse ist eine integrierte Entwicklungsumgebung, die unter professionellen Programmierern recht beliebt ist, denn Eclipse ist kostenfrei (für viele Unternehmen und Programmierer das wichtigste Argument überhaupt), recht leistungsstark und vor allem sehr flexibel. Eclipse kann nämlich durch Installation entsprechender Plug-Ins für die Erstellung praktisch beliebiger Anwendungen in beliebigen Sprachen – und dank des Android-Plug-Ins eben auch für die Erstellung von Android-Apps – hergerichtet werden.

DVD

Linux- und Windows-Anwender finden auf der Buch-DVD passende Setupdateien für Eclipse (Version Helios, geeignet für Android-SDK 11 und höher). Die jeweils aktuelle Version können Sie von der Eclipse-Website herunterladen: `http://www.eclipse.org/downloads`.

45.2.1 Installation

Um Eclipse zu installieren, müssen Sie lediglich die komprimierte Eclipse-Datei von der Buch-DVD extrahieren.

Wenn Sie bereits Eclipse in der Version Helios oder höher nutzen, können Sie diesen Schritt überspringen und direkt zur Installation des Android-Plug-Ins übergehen. Wenn Sie möchten, können Sie sich vorab aber noch einen eigenen Workspace für Ihre Android-Apps anlegen.

Deinstallation

Löschen Sie einfach das Verzeichnis der Eclipse-Installation.

45.2.2 Erster Start und Konfiguration

1. Zum Starten von Eclipse rufen Sie einfach die Datei *eclipse.exe* aus dem Extraktionsverzeichnis (= Installationsverzeichnis) auf.

Wenn Sie Eclipse das erste Mal starten, erscheint ein Dialogfeld, welches Sie nach dem zu verwendenden Workspace-Verzeichnis fragt. Die Vorgabe ist üblicherweise ein Verzeichnis namens *workspace*, das unter Ihrem Benutzerverzeichnis (unter Windows 7: *C:\Users\<Benutzername>*) angelegt wird.

2. Ersetzen Sie den vorgegebenen Pfad z.B. durch *C:\Projekte\Apps* und aktivieren Sie die Option *Use this as the default …*, um nicht bei jedem Start von Eclipse nach dem Workspace-Verzeichnis gefragt zu werden.
3. Klicken Sie auf OK.

EXKURS

Workspace (Arbeitsverzeichnis)

In Eclipse werden Programme (inklusive Apps), an denen Sie arbeiten, in Form von Projekten verwaltet. Dabei werden alle Dateien, die zu einem Projekt gehören (wie Quelltextdateien, Bilddateien usw.), in einem eigenen Verzeichnis abgelegt, dem sogenannten Projektverzeichnis, welches Sie angeben, wenn Sie ein neues Projekt anlegen.

*Die Projekte, die Sie anlegen, werden selbst wieder in übergeordneten Arbeitsverzeichnissen, den sogenannten **workspaces**, abgelegt. Wenn Sie den Workspace wie oben beschrieben beim ersten Start von Eclipse festlegen, werden alle Projekte, die Sie später anlegen, unter diesem Workspace abgelegt.*

Wenn Sie wieder beim Start von Eclipse nach dem zu verwendenden Workspace gefragt werden möchten, rufen Sie in Eclipse den Befehl Window/Preferences *auf, wechseln Sie zur Seite* General/Startup and Shutdown/Workspaces *und aktivieren Sie die Option* Prompt for workspace on startup.

Wenn Sie weitere Workspaces anlegen möchten oder den Workspace wechseln möchten, rufen Sie den Befehl File/Switch Workspace *auf.*

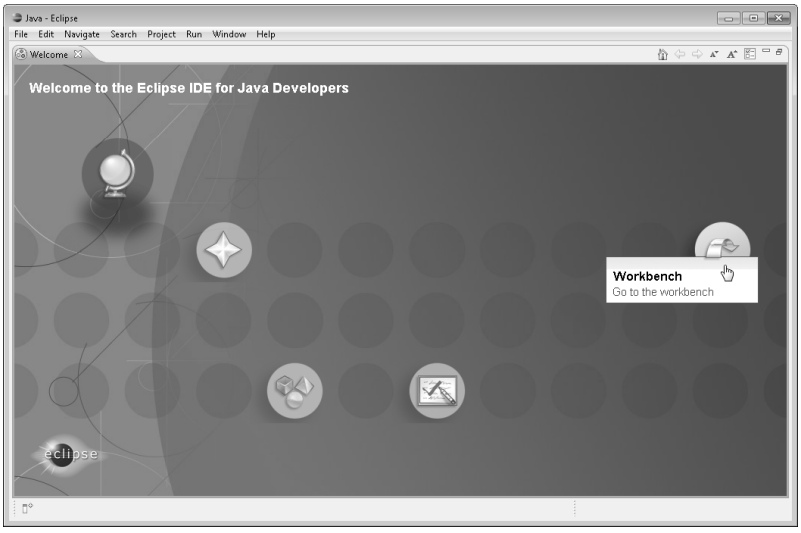


Abbildung 45.4: Eclipse wurde das erste Mal gestartet.

4. Klicken Sie im Begrüßungsbildschirm auf das Symbol Workbench.

TIPP

Legen Sie eine Desktop-Verknüpfung zu Eclipse an oder heften Sie die Datei *eclipse.exe* an das Startmenü an (wird unter Windows 7 als Menübefehl im Kontextmenü der Datei angeboten).

Java 6 Compliance einschalten

Im nächsten Schritt werden Sie das Android-Plug-In installieren. Das Plug-In, das bei Drucklegung dieses Buches angeboten wurde (Version 12), war allerdings noch nicht in der Lage, Class-Dateien im Java 7-Format zu verarbeiten. Sofern Sie also kein neueres Plug-In verwenden, das zu Java 7 kompatibel ist, müssen Sie Eclipse so einstellen, dass es beim Kompilieren Class-Dateien im Java 6-Format erzeugt. Für Eclipse-Helios – die Version, die Sie auf der Buch-DVD finden – sollte dies automatisch der Fall sein. Sie sollten es aber zur Sicherheit überprüfen.

5. Rufen Sie den Menübefehl WINDOW/PREFERENCES auf. Wählen Sie links die Seite JAVA/COMPILER aus und stellen Sie sicher, dass unter COMPILER COMPLIANCE LEVEL der Eintrag 1.6 ausgewählt ist.

45.3 Das Android-Plug-In

Jetzt müssen Sie nur noch unter Eclipse das Android-Plug-In installieren.

45.3.1 Installation

1. Falls noch nicht geschehen, starten Sie Eclipse und stellen Sie eine Internetverbindung her.
2. Rufen Sie den Menübefehl HELP/INSTALL NEW SOFTWARE auf.
   Es erscheint der Dialog AVAILABLE SOFTWARE.

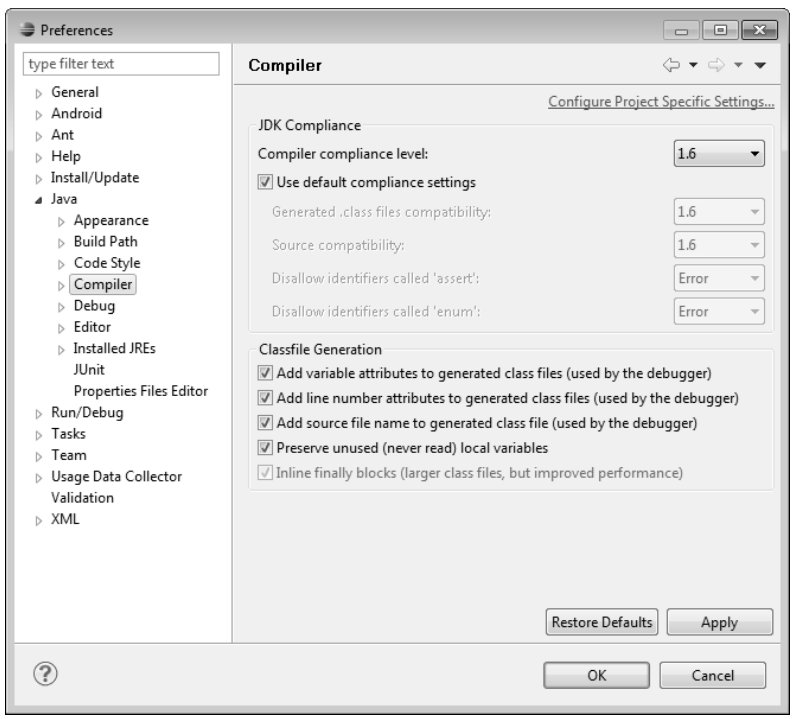


Abbildung 45.5: Einstellung der Java-Compliance

Das Dialogfeld ist anfangs leer und unbrauchbar. Um es zum Leben zu erwecken, müssen Sie eine Repository-Site einrichten.

3. Klicken Sie dazu rechts oben auf die Schaltfläche ADD.
   Es erscheint das Dialogfeld ADD REPOSITORY.

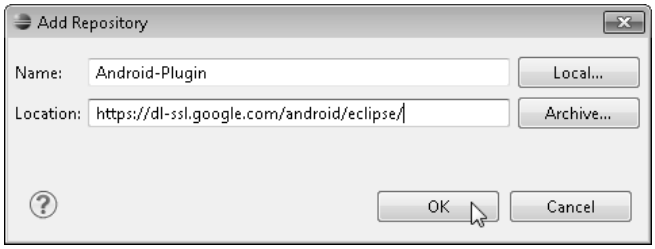


Abbildung 45.6: Geben Sie die Download-Adresse für das Android-Plug-In ein.

4. Geben Sie in das Feld NAME einen Titel für die Repository-Site ein, z.B.: »Android-Plug-In«.
   Der genaue Name spielt keine Rolle, sollte aber einen Hinweis darauf geben, welche Art von Software dahinter zu finden ist.
5. Geben Sie in das Feld LOCATION die nachfolgende Adresse ein und schicken Sie das Dialogfeld anschließend mit OK ab.

```
https://dl-ssl.google.com/android/eclipse/
```

Nach Abschicken des Dialogs greift Eclipse auf die angegebene Site zu und zeigt das vorgefundene Angebot an. Im Dialogfeld sollte nun der Eintrag DEVELOPER TOOLS zu lesen sein.

6. Aktivieren Sie das Kontrollkästchen neben DEVELOPER TOOLS und klicken Sie auf die Schaltfläche NEXT.

Es erscheint das Dialogfeld INSTALL DETAILS, in dem die zu installierenden Komponenten noch einmal zusammengefasst sind. Achten Sie darauf, dass sich darunter ein Eintrag für die Android Development Tools, kurz ADT, befindet.

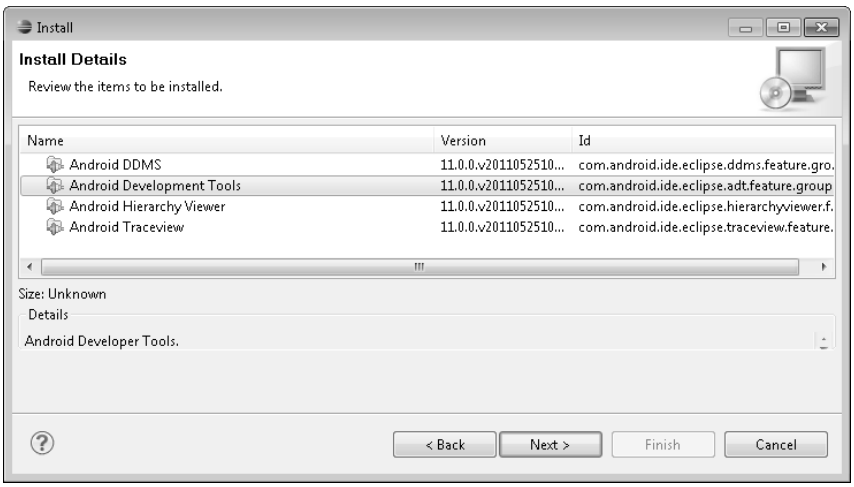


Abbildung 45.7: Die Liste der zu installierenden Komponenten

7. Klicken Sie auf NEXT, um die Komponenten installieren zu lassen.

Höchstwahrscheinlich werden Sie vorab Lizenzbedingungen und vermutlich auch das Installieren von nicht signierter Software bestätigen müssen.

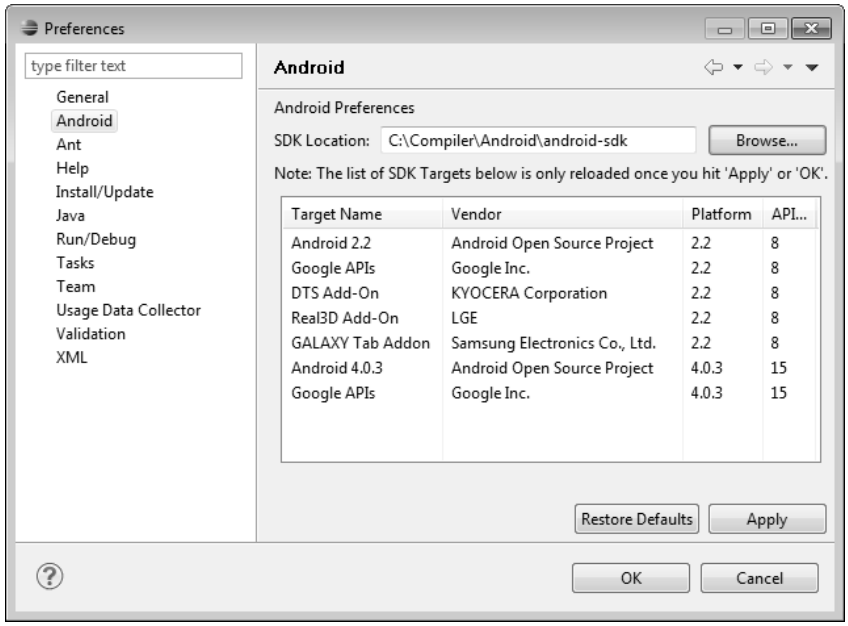
8. Zum guten Schluss müssen Sie Eclipse neu starten.

45.3.2 Konfiguration

Jetzt müssen Sie Eclipse nur noch mitteilen, wo es das Android-SDK findet.

1. Starten Sie, sofern nicht schon geschehen, Eclipse.
2. Rufen Sie den Menübefehl WINDOW/PREFERENCES auf.
   Es erscheint der Dialog PREFERENCES.
3. Wählen Sie links die Kategorie ANDROID aus.
4. Klicken Sie auf BROWSE und wählen Sie in dem erscheinenden Dialog das Installationsverzeichnis Ihres Android-SDK aus.
5. Klicken Sie dann auf APPLY …
   Für jede der Android-Plattformen, die Sie mit dem SDK heruntergeladen und installiert haben, wird nun ein Target (Ziel für die App-Erstellung) eingerichtet.
6. … und anschließend auf OK.

Abbildung 45.8:
Der letzte Klick



Glückwunsch! Ihr Rechner ist nun fertig eingerichtet.

45.4 Die erste App

Die App, die wir in diesem Kapitel schreiben werden, soll nichts weiter tun, als uns auf dem Smartphone mit einem freundlichen »Hallo Programmierer« zu begrüßen. Das ist nicht gerade viel, aber es geht auch gar nicht darum, im ersten Versuch gleich eine sinnvolle und perfekte App zu erstellen. Es geht darum, einen Überblick über den App-Entwicklungsprozess zu bekommen.

45.4.1 Das Projekt

Wer Eclipse als Entwicklungsumgebung verwendet, für den beginnt die Arbeit an jeder neuen App mit dem Anlegen eines passenden Projekts. Starten Sie also Eclipse und legen Sie ein neues Projekt an.

Projekte, Projekttypen und Wizards

EXKURS

Projekte sind die Eclipse-interne Verwaltungseinheit, in der Eclipse alle Daten und Dateien zusammenfasst, die nötig sind, um aus einem Quelltext eine App zu erstellen (oder auch ein Java-Konsolenprogramm, eine C++-Programmbibliothek oder was auch immer Sie programmieren möchten).

*Unterschiedliche Programmierziele – möchten Sie eine App schreiben, ein normales Java-Programm oder »nur« eine Klassenbibliothek? – erfordern unterschiedlich konfigurierte Projekte. Um Ihnen die Konfiguration zu erleichtern, gibt es in Eclipse die sogenannten **Wizards**: dialogbasierte Assistenten, die darauf spezialisiert sind, Ihnen bei der Einrichtung von Projekten eines bestimmten Typs zu helfen. Welche Wizards Ihnen im Einzelnen zur Verfügung stehen, hängt davon ab, welche Plug-Ins Sie installiert haben. Für die App-Programmierung sollte jedenfalls nach erfolgreicher Installation des Android-Plug-Ins ein passender Wizard vorhanden sein.*

1. Rufen Sie im Menü den Befehl FILE/NEW/PROJECT auf.
2. Wählen Sie im Dialogfeld NEW PROJECT unter dem Ordner ANDROID den Eintrag ANDROID PROJECT und klicken Sie auf NEXT.

Die Anordnung der Befehle im Eclipse-Menüsystem kann je nach Eclipse-Version und installierten Plug-Ins variieren.

INFO

Es erscheint das Dialogfeld NEW ANDROID PROJECT.

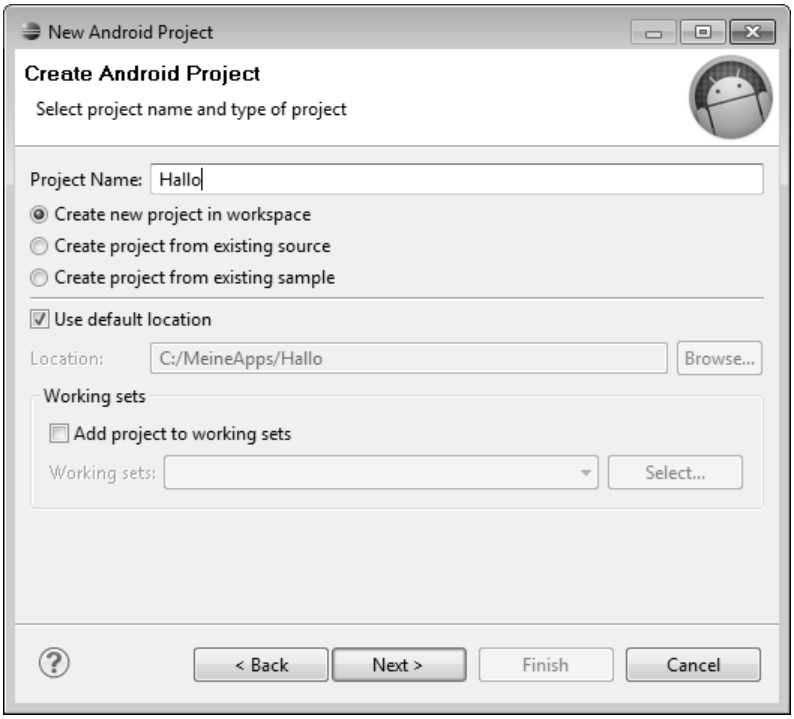


Abbildung 45.9: Die erste Seite des Dialogfelds zum Anlegen neuer Projekte

3. Geben Sie als Namen für das Projekt *Hallo* ein und klicken Sie auf NEXT.

   Unter diesem Namen wird Ihr Projekt fortan in Eclipse aufgeführt.

   Außerdem legt Eclipse auf der Festplatte, unter dem Verzeichnis des aktuellen Workspace ein Verzeichnis dieses Namens als Projektverzeichnis an. Jedenfalls dann, wenn Sie das Kontrollkästchen USE DEFAULT LOCATION aktiviert lassen.

   Verwenden Sie für den Projektnamen keine Sonderzeichen und möglichst auch keine Umlaute oder Leerzeichen.

   Achten Sie darauf, dass die Option CREATE NEW PROJECT IN WORKSPACE aktiviert ist, damit Eclipse ein neues Projektverzeichnis unter dem Verzeichnis des aktuellen Workspace anlegt.

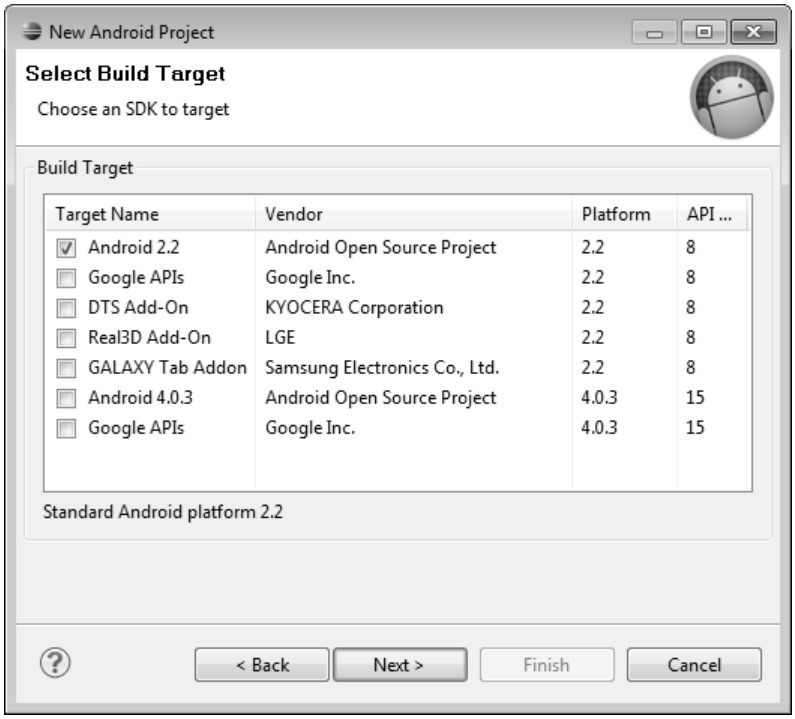


Abbildung 45.10: Die Zielplattform auswählen

4. Auf der nächsten Seite des Dialogfelds markieren Sie die gewünschte Zielplattform. Für unsere Programme ist dies immer Android 2.2.

   Das Build Target gibt an, für welche Android-Version (Target) Eclipse Ihr Projekt später erstellen (Build) soll.

   Wir arbeiten in diesem Buch standardmäßig mit der Version 2.2, da diese einen guten Kompromiss aus Leistungsfähigkeit und Verbreitung darstellt, siehe auch Kasten.

   Welche Build Targets im Dialogfeld aufgelistet werden, hängt übrigens davon ab, welche Plattformen Sie mit dem SDK-Manager heruntergeladen haben.

5. Nun muss noch das dritte und letzte Dialogfeld gefüllt werden. Da die Einstellungen etwas umfangreicher sind, haben wir sie in Tabelle 1 zusammengefasst.

   Da die Einstellungen etwas umfangreicher sind, haben wir sie in Tabelle 1 zusammengefasst.

Tabelle 45.2: Einstellungen für das New Android Project-Dialogfeld

Feld	Eingabe	Bedeutung
APPLICATION NAME	*Hallo*	Unter diesem Namen wird Ihre App später auf dem Smartphone geführt.
PACKAGE NAME	*ihrname.hallo*	Die Angabe ist für App-Projekte zwingend erforderlich und muss eindeutig sein, um sicherzustellen, dass es auf einem Android-System keine zwei Apps mit gleichem Paketnamen gibt. Typisch ist daher die Verwendung von Domainnamen in umgekehrter Reihenfolge mit angehängtem App-Namen (also z.B. *de.meinewebsite.hallo*). Solange Sie aber nur Apps für den Eigenbedarf schreiben, sollte eine Kombination aus Ihrem Namen und dem Namen der App genügen. (Wir verwenden in unseren Beispielprojekten den Namen *de.carpelibrum* plus dem jeweiligen App-Namen.)
CREATE ACTIVITY	*HalloActivity*	Apps definieren üblicherweise eine oder mehrere Activities, die ausgeführt werden können. Wenn Sie das Kontrollkästchen CREATE ACTIVITY aktivieren, legt Eclipse für die neue App gleich eine Activity (Aktivität) des vorgegebenen Namens an. Was diese Activity macht, müssen Sie aber natürlich später selbst programmieren. Achtung! Der Name darf keine Leer- und keine Sonderzeichen enthalten, da das Android-Plug-In unter diesem Namen eine Klasse definiert.
MIN SDK VERSION	*8*	Die Mindestversion des SDK muss zu dem ausgewählten Build Target passen. Wenn Sie das Feld freilassen, wird automatisch die dem Target entsprechende SDK-Version ausgewählt (also 8 für Android 2.2). Wenn Sie die Mindest-SDK-Version explizit angeben, darf diese allerdings nicht größer sein als die API-Version, die dem ausgewählten Target entspricht (siehe Tabelle in Dialogfeld).

STOP Wenn Sie Namen für Verzeichnisse oder Programmelemente vergeben, verzichten Sie auf Leerzeichen, Sonderzeichen und möglichst auch auf Umlaute (obwohl letztere mittlerweile von den meisten Systemen und Programmiersprachen korrekt verarbeitet werden).

6. Klicken Sie auf FINISH, um das Projekt anlegen zu lassen.
   Wenn Sie zuerst noch einmal auf NEXT klicken, erhalten Sie die Möglichkeit, zu dem neuen Projekt auch gleich noch ein passendes Testprojekt anzulegen. In diesem Buch werden wir von dieser Möglichkeit allerdings keinen Gebrauch machen.

INFO Das *New Android Project*-Dialog verfügt über eine automatische Fehlerkontrolle. Falsche oder widersprüchliche Einstellungen werden von dieser erkannt und oben im Dialogfeld angezeigt.

Das Projekt in Eclipse

Zurück im Hauptbildschirm von Eclipse sehen Sie das neu angelegte Projekt. Anfänglich nur als einzelner unscheinbarer Eintrag im *Package Explorer*, doch

wenn Sie auf den Namen doppelklicken, wird der Projektknoten expandiert und es zeigt sich, dass das Projekt bereits aus einer umfangreichen Sammlung von Ordnern und Dateien besteht.

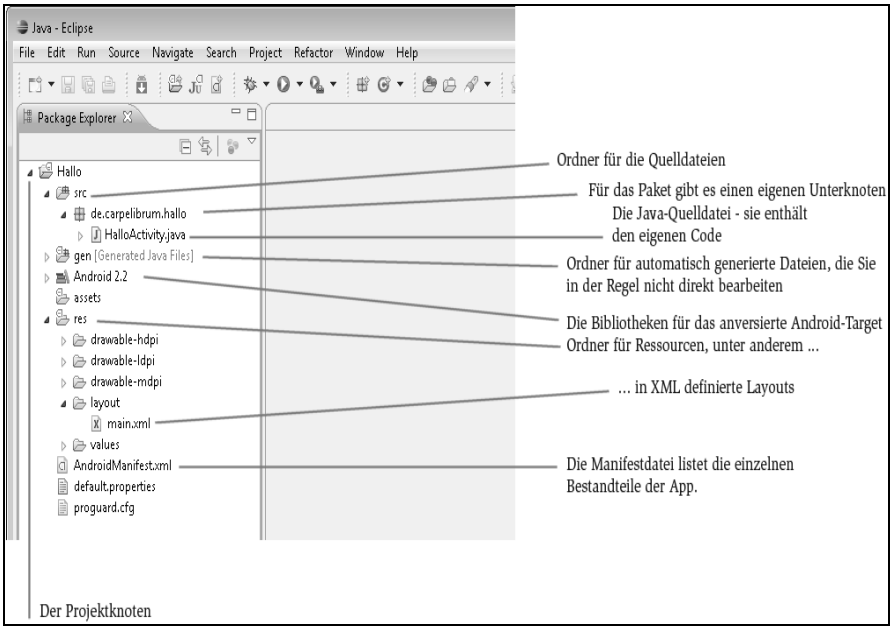


Abbildung 45.11: Das neu angelegte Projekt

TIPP

Falls Sie den Package Explorer aus Versehen geschlossen haben oder er aus irgendeinem anderen Grund nicht angezeigt wird, können Sie ihn jederzeit durch Aufruf des Menübefehls WINDOW/SHOW VIEW/PACKAGE EXPLORER wieder anzeigen lassen.

45.4.2 Das vorgegebene Codegerüst

Werfen wir einen Blick in das Codegerüst, das bereits für uns erzeugt wurde.

1. Doppelklicken Sie im Package Explorer auf den Knoten der Quelltextdatei *HalloActivity.java*, um die Datei in den Editor zu laden.

Listing 45.1: Inhalt der Datei HalloActivity.java

```
package de.carpelibrum.hallo;

import android.app.Activity;
import android.os.Bundle;

public class HalloActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

In der obersten Zeile treffen wir zunächst den Paketnamen wieder, den wir im Dialogfeld NEW ANDROID PROJECT eingegeben haben.

Unter der Paketangabe folgen `import`-Anweisungen für die Klassen `Activity` und `Bundle` aus der Android-Bibliothek. Die Klasse `Bundle` ist ein Typ, der zur Übergabe von Statusinformationen benutzt wird. Die Klasse `Activity` ist die Basisklasse für Activities, also auch für unsere `HalloActivity`-Klasse:

```
public class HalloActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Rekapitulieren wir kurz, was wir bereits über Activities erfahren haben, und ergründen wir, wie weit sich dies in dem obigen Code niederschlägt:

- Jede Activity erfüllt eine bestimmte Aufgabe.

  Aufgabe der `HalloActivity` soll es sein, uns mit dem Text »Hallo Programmierer« zu begrüßen. Für diese einfache Aufgabe ist kein besonderer Code erforderlich. Die Activity wird einfach auf ihrer Bildschirmseite ein entsprechendes Textfeld anzeigen (siehe nachfolgenden Abschnitt).
- Activities unterliegen einem in mehrere Phasen eingeteilten Lebenszyklus. Durch Überschreiben spezieller Activity-Methoden kann der Programmierer festlegen, was in diesen Phasen geschehen soll.

  Eine diese Methoden ist im Grundgerüst bereits überschrieben worden: `onCreate()`.
- Activities sind eine Kombination aus Bildschirmseite und zugehörigem Code.

  Das statische Layout einer Bildschirmseite wird in Android, wie bereits erwähnt, üblicherweise nicht durch Code, sondern durch eine XML-Layoutdatei definiert.

  Für unsere `HalloActivity` wurde bereits eine Layoutdatei angelegt. Sie heißt *main.xml* und liegt im Verzeichnis *res/layout*. Gleichzeitig wurde in der Datei *R.java* eine Ressourcen-ID `R.layout.main` definiert, die mit der Layoutdatei verbunden ist.

  Damit erklärt sich die Bedeutung der zweiten Anweisung der `onCreate()`-Methode. Der Aufruf von `setContentView(R.layout.main)` verbindet die Activity mit der in *main.xml* definierten Benutzeroberfläche.

Activity-Klassen müssen immer `public` sein, damit sie vom System instanziiert werden können.

INFO

R ist der Name einer Klasse, die automatisch für das Projekt erzeugt wird und deren Aufgabe die **R**essourcenverwaltung ist (daher der Name R). Diese Klasse enthält als Elemente eine Reihe von untergeordneten Klassen, darunter eben auch die Klasse layout, in der die ID main als statische Konstante definiert ist.

45.4.3 Bildschirmseite und Layoutdatei

Aus Sicht des Programmierers ist eine Bildschirmseite eine Hierarchie von View-Objekten.

Ganz oben in der Hierarchie steht üblicherweise eine *Layout-View* – eine spezielle Art von View, die andere untergeordnete View-Elemente einbetten kann und nach bestimmten Layoutregeln anordnet. Layout-Views sind von der Basisklasse ViewGroup abgeleitet (welche ihrerseits von View abgeleitet ist).

Eingebettet in die oberste Layout-View folgen wahlweise weitere Layout-Views und/oder einfache Views für Steuerelemente wie Schaltflächen, Listenfelder usw. – auch *Widgets* genannt.

Obwohl es wie bereits erwähnt auch möglich ist, die View-Hierarchie einer Bildschirmseite per Code zu erzeugen, ist es in der Android-Programmierung üblich, das statische Layout einer Bildschirmseite per XML zu definieren.

Für die Bildschirmseite zu unserer HalloActivity ist dies dankenswerter Weise bereits geschehen.

1. Expandieren Sie im Package Explorer den Knoten *res/layout* und doppelklicken Sie auf den Eintrag für die Datei *main.xml*, um diese in den Editor zu laden.
   Für XML-Layoutdateien gibt es zwei unterschiedliche Ansichten, die über Reiter am unteren Rand des Editors ausgewählt werden. Per Voreinstellung zeigt Eclipse immer erst die Ansicht GRAPHICAL LAYOUT.
2. Klicken Sie am unteren Rand des Editors auf den Reiter MAIN.XML, um die Layoutdatei in der XML-Ansicht zu betrachten.

Listing 45.2: Inhalt der Layoutdatei main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">

   <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/hello" />

</LinearLayout>
```

Die oberste View bildet in diesem Fall eine LinearLayout-View.

LinearLayout ordnet eingebettete Views entweder neben- oder untereinander an – je nachdem, ob dem Attribut `android:orientation` der Wert `"horizontal"` oder `"vertical"` zugewiesen wurde.

Der Wert `"fill_parent"`, der den Attributen `android:layout_width` und `android:layout_height` zugewiesen wurde, sorgt dafür, dass die LinearLayout-View den übergeordneten Container (in diesem Fall der Anzeigebereich des Touchscreens des Android-Geräts) in Breite und Höhe ganz ausfüllt.

In die LinearLayout-View wurde eine einzige untergeordnete View eingebettet: eine TextView. TextView-Steuerelemente dienen dazu, einen Text anzuzeigen.

Der anzuzeigende Text kann dem Attribut `android:text` direkt zugewiesen werden. Er kann aber auch – und dies ist der übliche Weg – als String-Ressource definiert und dann über seine Ressourcen-ID zugewiesen werden. Letzteres ist im obigen Code der Fall.

Ansonsten ist die TextView so konfiguriert, dass sie sich über die gesamte Breite der übergeordneten LinearLayout-View erstreckt, in ihrer Höhe aber nach dem enthaltenen Text und der verwendeten Schriftgröße richtet.

STOP

Für jedes dieser XML-Elemente wird bei der Kompilierung die entsprechende Android-Klasse instanziiert und das so erzeugte Objekt gemäß den in den XML-Elementen definierten Attributen konfiguriert.

45.4.4 Ressourcen

In der App-Programmierung ist es üblich, Texte und Bildelemente, die auf der Benutzeroberfläche angezeigt werden, als Ressourcen zu verwalten. Für Textelemente gibt es dazu z.B. die Datei *strings.xml*.

3. Expandieren Sie im Package Explorer die Knotenfolge *res/values* und doppelklicken Sie auf den Eintrag für die Datei *strings.xml*, um diese in den Editor zu laden.

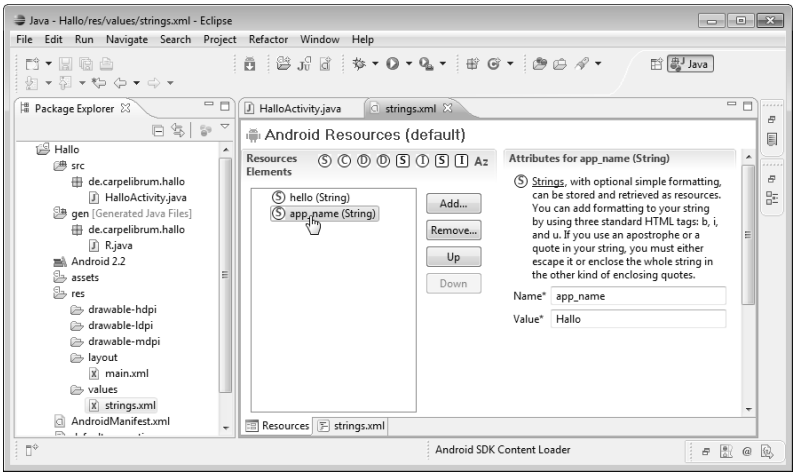


Abbildung 45.12: Die Datei strings.xml im Ressourceneditor

In Eclipse können Sie die Ressourcendatei *strings.xml* wahlweise über die Eingabemaske des Ressourceneditors (siehe Abbildung 45.12) oder als reinen XML-Code bearbeiten.

Um zwischen Ressourceneditor und Texteditor hin und her zu schalten, klicken Sie einfach am unteren Rand des Editorfenster auf die zugehörigen Reiter.

Listing 45.3: Inhalt der Datei strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HalloActivity!</string>
    <string name="app_name">Sag Hallo</string>
</resources>
```

Um den auszugebenden Text von »Hello World, HalloActivity!« in »Hallo Programmierer!« zu ändern, geben Sie den neuen Text einfach zwischen den Tags des ersten `string`-Elements ein:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hallo Programmierer!</string>
    <string name="app_name">Hallo</string>
</resources>
```

4. Speichern Sie Ihre Änderungen, indem Sie den Menübefehl FILE/SAVE ALL aufrufen.

45.4.5 Die App erstellen (Build)

Eclipse ist standardmäßig so eingestellt, dass Änderungen an einem Projekt sofort kompiliert und ausgewertet werden. Dies ist bequem, kann aber auch manchmal nerven, vor allem, wenn ungewiss bleibt, ob es mit der Erstellung geklappt hat oder nicht. Lesern, die mit Eclipse noch nicht so vertraut sind, empfehlen wir daher, die Option vorerst zu deaktivieren und ihre Projekte manuell zu erstellen.

1. Öffnen Sie das PROJECT-Menü.
2. Wenn vor dem Befehl BUILD AUTOMATICALLY ein Häkchen zu sehen ist, wählen Sie den Befehl aus, um die Option zu deaktivieren und das Häkchen verschwinden zu lassen.

Um Ihr Projekt von Hand zu erstellen, gehen Sie wie folgt vor:

1. Speichern Sie das Projekt mit dem Menübefehl FILE/SAVE ALL (oder durch Klick auf die entsprechende Symbolschaltfläche).
2. Rufen Sie den Menübefehl PROJECT/BUILD PROJECT auf.
   Alternativ können Sie im Package Explorer von Eclipse mit der rechten Maustaste auf den Projektknoten HALLO klicken und im Kontextmenü den Befehl BUILD PROJECT aufrufen.

EXKURS

Fehlerkorrektur in Eclipse

Gibt es Fehler im Projekt, markiert Eclipse die betroffenen Stellen im Editor (rote Wellenlinie und Lampensymbol mit rotem Kästchen in Randleiste). Ob Code betroffen ist, der gerade nicht im Editor angezeigt wird, können Sie im Package Explorer kontrollieren. Enthält eine Datei Fehler, werden die Datei und alle übergeordneten Ordner mit einem roten Kästchen mit weißem Kreuz markiert. (Warnungen werden gelb markiert.)

Oft bietet das QuickInfo-Fenster unter dem Fehlertext einen oder mehrere Optionen an, wie der Fehler behoben werden könnte. Diese Korrekturvorschläge führen allerdings nicht immer zum gewünschten Ziel! Klicken Sie nur dann auf eine der angebotenen Optionen, wenn Sie sicher sind, dass der Fehler damit in Ihrem Sinne behoben wird.

Fehler, die bei der Erstellung auftreten, werden auch im Fenster PROBLEMS (Aufruf über Menübefehl WINDOW/SHOW VIEW/PROBLEMS) aufgelistet.

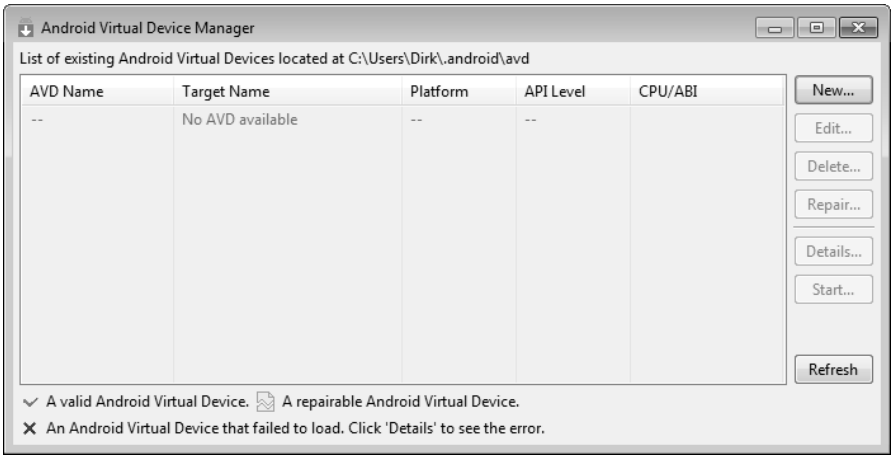
45.5 Die App im Emulator testen

Um Apps bequem und effizient testen zu können, gibt es im Android SDK einen Emulator, mit dessen Hilfe Sie auf Ihrem Rechner unterschiedliche Android-Geräte simulieren können. Für jeden Gerätetyp müssen Sie allerdings zuvor ein »Android Virtual Device« (kurz AVD) anlegen.

AVD für Emulator anlegen

1. Starten Sie den Android-Manager.
   Sie können den SDK-Manager direkt von Eclipse aus starten, indem Sie auf sein Symbol in der Symbolleiste klicken.

Abbildung 45.13: Anlegen eines Emulators (Virtual Device) im SDK-Manager



2. Klicken Sie auf die Schaltfläche NEW.
   Es erscheint das Dialogfeld CREATE NEW ANDROID VIRTUAL DEVICE.
3. Geben Sie als Namen für das AVD *AVD_2.2_800* ein.
   Der Name sollte Auskunft über die wichtigsten Konfigurationsparameter wie Zielplattform (hier 2.2) und Touchscreen-Auflösung (hier WVGA800) geben.
4. Wählen Sie als Target die Plattform *Android 2.2* aus.
   Das Listenfeld führt alle Plattformen auf, für die Sie zuvor mit dem SDK-Manager die SDK heruntergeladen und installiert haben.
5. Behalten Sie unter SKIN die Einstellung BUILT-IN mit der Auswahl WVGA800 bei.
6. Drücken Sie auf CREATE AVD.
7. Schließen Sie den AVD-Manager.

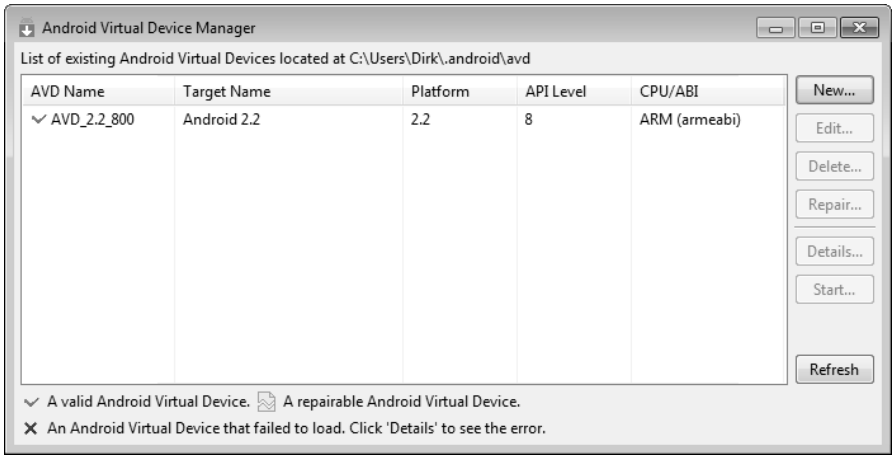


Abbildung 45.14: Das AVD ist eingerichtet.

INFO

Zum Einarbeiten in die App-Programmierung genügt in der Regel ein einziges AVD. Sobald Sie aber beginnen, Apps an Bekannte weiterzugeben oder gar auf dem Android Market Place anzubieten, sollten Sie mehrere AVDs für unterschiedliche Plattformen (Einstellung TARGET), Auflösungen (Einstellung SKIN) und Hardware-Ausstattungen (Einstellungen SD CARD und HARDWARE) einrichten, um bequem testen zu können, wie sich Ihre App auf unterschiedlichen Android-Geräten verhält.

App testen

1. Rufen Sie den Befehl WINDOW/SHOW VIEW/CONSOLE auf, um das CONSOLE-Fenster einzublenden.
   In diesem Fenster werden die Statusnachrichten und Meldungen des Emulators angezeigt.
2. Rufen Sie den Befehl RUN/RUN auf, um die App zu testen.
   Alternativ können Sie auch im Package Explorer von Eclipse mit der rechten Maustaste auf den Projektknoten HALLO klicken und im Kontextmenü den Befehl RUN AS/ANDROID APPLICATION aufrufen.

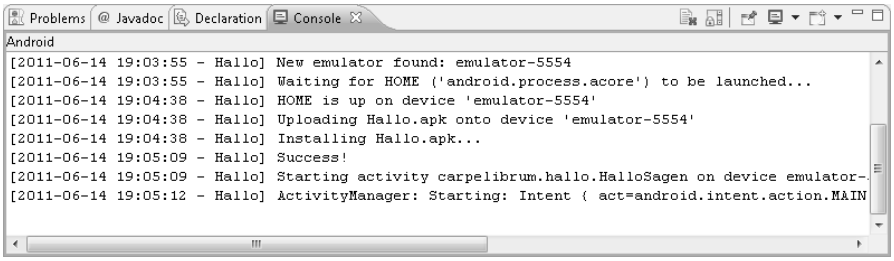
STOP

Wenn Sie ein Projekt das erste Mal mit dem Befehl Run/Run ausführen, erscheint das Dialogfeld Run As, in dem Sie Android Application als Startmodus auswählen müssen. Eclipse erstellt daraufhin eine passende Launch-Konfiguration und benutzt diese fortan automatisch.

Im Console-Ausgabefenster von Eclipse erscheinen nun nacheinander diverse Meldungen und auf Ihrem Bildschirm sehen Sie das Fenster des Emulators.

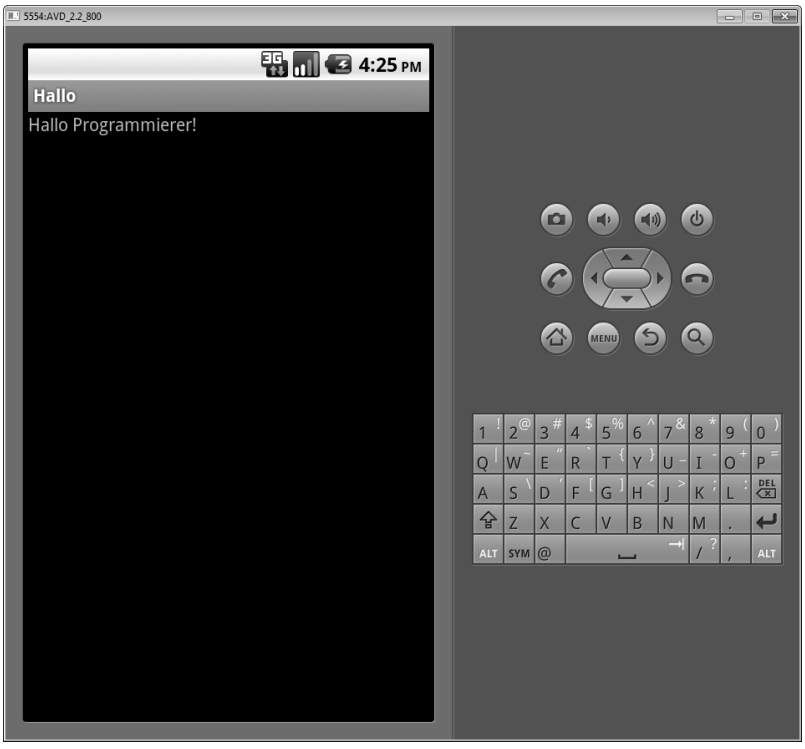
3. Warten Sie, bis im Console-Fenster die Meldung »ActivityManager: Starting: Intent ...« erscheint.

Abbildung 45.15: Die App ist im Emulator und gestartet.


Problems
Javadoc
Declaration
Console
Android
[2011-06-14 19:03:55 - Hallo] New emulator found: emulator-5554
[2011-06-14 19:03:55 - Hallo] Waiting for HOME ('android.process.acore') to be launched...
[2011-06-14 19:04:38 - Hallo] HOME is up on device 'emulator-5554'
[2011-06-14 19:04:38 - Hallo] Uploading Hallo.apk onto device 'emulator-5554'
[2011-06-14 19:04:38 - Hallo] Installing Hallo.apk...
[2011-06-14 19:05:09 - Hallo] Success!
[2011-06-14 19:05:09 - Hallo] Starting activity carpelibrum.hallo.HalloSagen on device emulator-
[2011-06-14 19:05:12 - Hallo] ActivityManager: Starting: Intent { act=android.intent.action.MAIN

4. Klicken Sie falls nötig im Emulator auf die Schaltfläche Menu.

Abbildung 45.16: Die Benutzeroberfläche unserer App – mit Titelleiste und darunter gelegenem Textfeld



Wie Sie sehen, wurde die `onCreate()`-Methode bereits ausgeführt und im Textfeld ist der von uns angepasste Ausgabetext zu lesen.

5. Beenden Sie die App, indem Sie im Emulator auf die ZURÜCK-Taste klicken.
6. Schließen Sie das Emulatorfenster.

TIPP

Solange Sie an einer App arbeiten, sollten Sie den einmal gestarteten Emulator nicht unnötig neu starten. Nicht, dass dies zu Problemen führen würde, es kostet nur sehr viel Zeit. Lassen Sie den Emulator also während der Arbeit an der App geöffnet und starten Sie das App einfach zum Testen mit dem Befehl RUN/RUN neu.

45.6 Die App auf dem Smartphone testen

Wenn Sie ein Android-Smartphone mit Android 2.2 oder höher besitzen, können Sie Ihre Apps zum Begutachten und Testen auch direkt von Eclipse aus auf das Smartphone übertragen.

Voraussetzung ist allerdings, dass Sie Ihr Smartphone per USB an den Rechner anschließen und für das USB-Debugging konfigurieren (Informationen hierzu finden Sie in der Regel im Benutzerhandbuch Ihres Smartphones). Für das Galaxy-Smartphone von Samsung müssen Sie z.B. unter EINSTELLUNGEN/ANWENDUNGEN/ENTWICKLUNG die Option USB-DEBUGGING aktivieren und die Kies-Treiber von der Begleit-DVD auf dem Rechner installieren.

INFO

Wenn Sie kein Smartphone besitzen oder die USB-Übertragung nicht klappt, ist dies nicht weiter tragisch. Für die Einarbeitung in die Android-Programmierung können Sie auch ausschließlich mit dem Emulator testen (siehe vorangehenden Abschnitt).

Automatische Übertragung mit Eclipse

Um Eclipse so einzurichten, dass beim Ausführen der App von Eclipse aus das gewünschte Zielgerät (Emulator oder angeschlossenes Smartphone) abgefragt wird, gehen Sie wie folgt vor

1. Stellen Sie Ihr Smartphone auf USB-Debugging um.
2. Stecken Sie das USB-Kabel ein.
3. Rufen Sie in Eclipse den Menübefehl RUN/RUN CONFIGURATIONS auf.
4. Wählen Sie, falls nötig, im linken Teil des Dialogfelds RUN CONFIGURATIONS unter der Kategorie ANDROID APPLICATION Ihre App (in unserem Fall *Hallo*) aus.
5. Wechseln Sie im rechten Teil zur Registerseite TARGET und wählen Sie als DEPLOYMENT TARGET SELECTION MODE die Option MANUAL aus.
6. Klicken Sie auf APPLY und anschließend auf RUN.

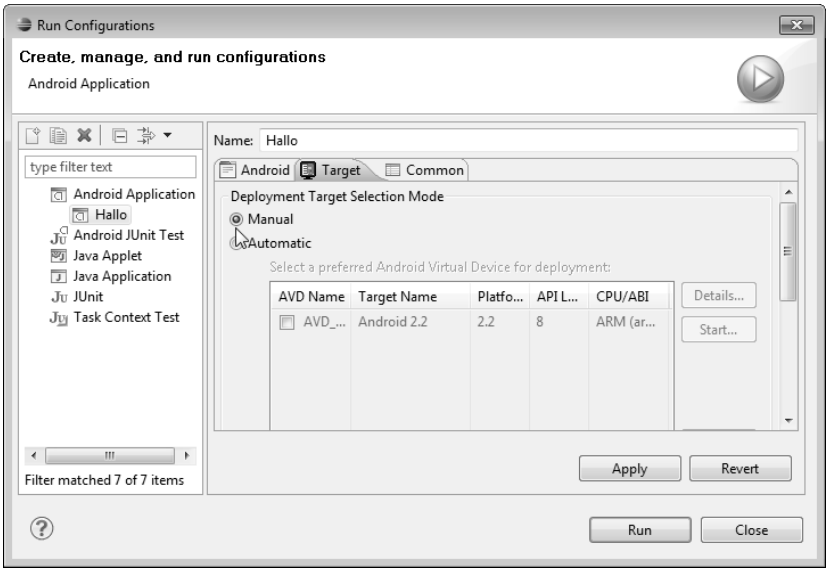


Abbildung 45.17: Eclipse-Konfiguration für die manuelle Auswahl des Zielgeräts

Danach erscheint das Dialogfeld ANDROID DEVICE CHOOSER, in dem Ihr angeschlossenes Smartphone unter seiner Seriennummer aufgeführt sein sollte.

7. Wählen Sie das Smartphone mit einem Mausklick aus und drücken Sie dann OK.

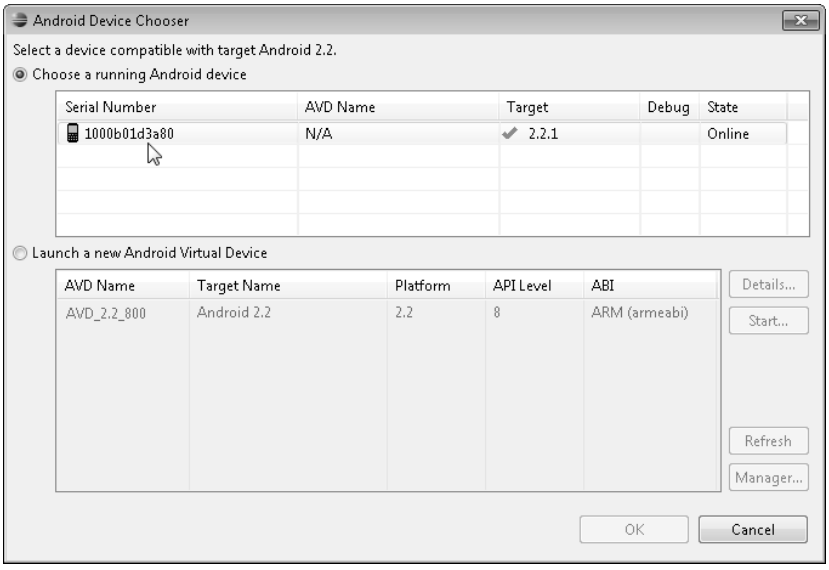


Abbildung 45.18: Auswahl des angeschlossenen Smartphones als Zielgerät

INFO

Das Dialogfeld ANDROID DEVICE CHOOSER wird fortan bei jeder Ausführung der App aufgerufen – so lange, bis Sie die Launch-Konfiguration wieder auf AUTOMATIC umstellen.